Lab Report - 7

Course code: CSE422

Course Title: Computer Graphics Lab

Submitted to:

Deawan Rakin Ahamed Remal

Lecturer, Department of CSE

Daffodil International University

Submitted by:

ISHAN AHMAD

203-15-14521

57-B

Department of CSE

Daffodil International University

Submission Date: 4th November, 2023

Experiment: 2D Transformation’s Implementation

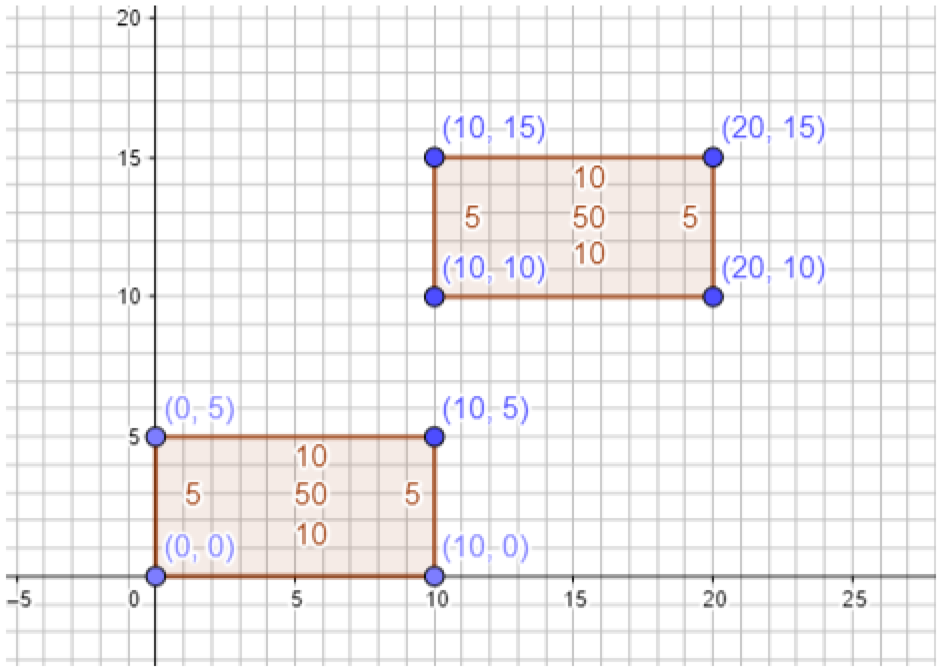
Task 1: Translation

Code:

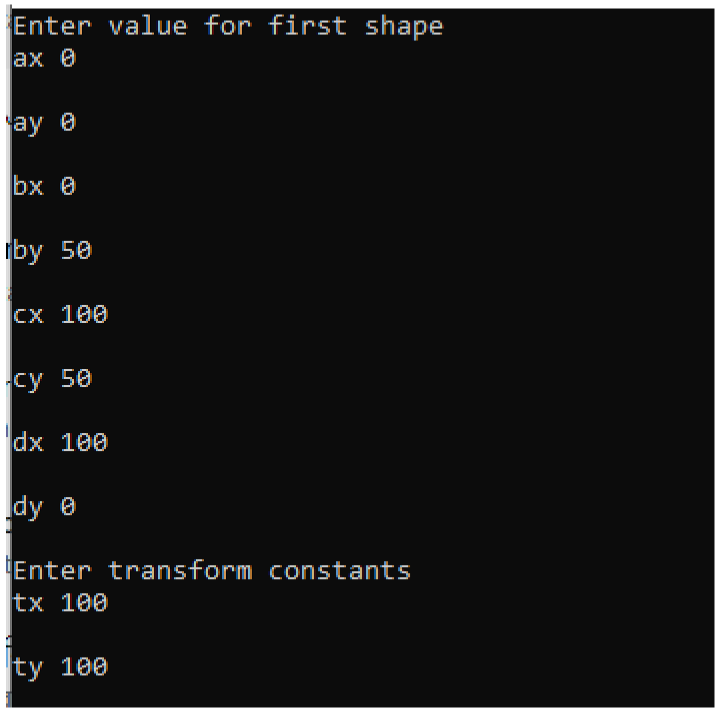
```
#include<windows.h>
#include <GL/glut.h>
#include <iostream>
using namespace std;

int ax, ay, bx, by, cx, cy, dx, dy, tx, ty;

void init(void)
{
     glClearColor(0.0, 0.0, 0.0, 0.0);

     glMatrixMode(GL_PROJECTION);

     gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
}

void drawShapes(void)
{
     glClear(GL_COLOR_BUFFER_BIT);

     glColor3f(1, 0, 0);
     glBegin(GL_QUADS);
       glVertex2i(ax, ay);
       glVertex2i(bx, by);
       glVertex2i(cx, cy);
       glVertex2i(dx, dy);
    glEnd();

    glColor3f(1, 1, 0);
     glBegin(GL_QUADS);
       glVertex2i(ax+tx, ay+ty);
       glVertex2i(bx+tx, by+ty);
       glVertex2i(cx+tx, cy+ty);
```

```
        glVertex2i(dx+tx, dy+ty);
    glEnd();

glFlush();

}
int main(int argc, char* argv[])
{
     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

     glutInitWindowPosition(100, 100);
     glutInitWindowSize(700, 700);
     glutCreateWindow("Translation");

     init();
     glutDisplayFunc(drawShapes);

     cout << "Enter value for first shape " << endl;
     cout << "ax ";
     cin >> ax;
     cout << endl;
     cout << "ay ";
     cin >> ay;
     cout << endl;

    cout << "bx ";
     cin >> bx;
     cout << endl;
     cout << "by ";
     cin >> by;
     cout << endl;

     cout << "cx ";
     cin >> cx;
     cout << endl;
     cout << "cy ";
     cin >> cy;
     cout << endl;
    cout << "dx ";
     cin >> dx;
     cout << endl;
     cout << "dy ";
     cin >> dy;
     cout << endl;

     cout << "Enter transform constants " << endl;
     cout << "tx ";
     cin >> tx;
     cout << endl;

     cout << "ty ";
     cin >> ty;
     cout << endl;

     glutMainLoop();

     return 0;
}
```

Graph:



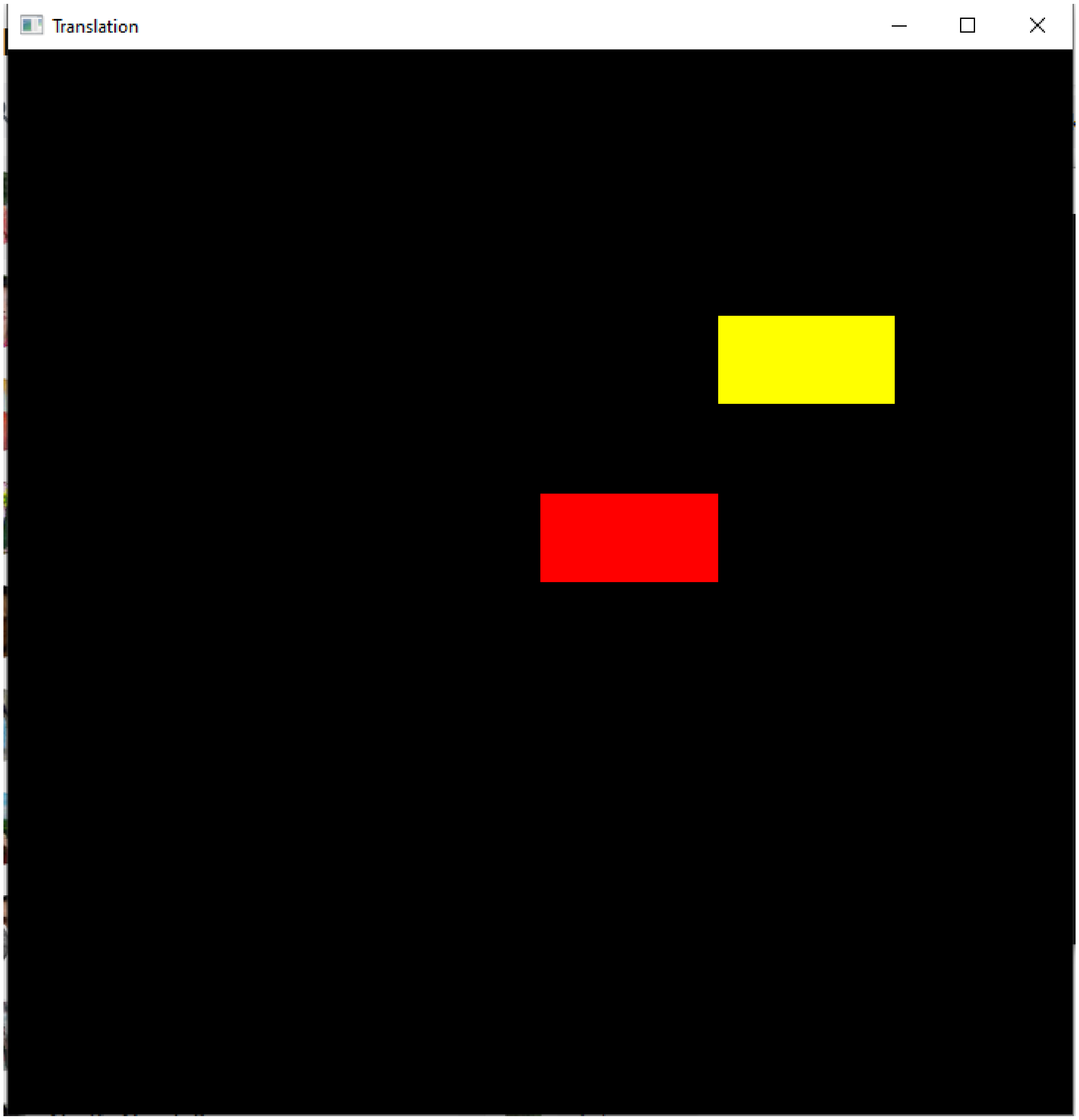
Input:



```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Enter transform constants
tx 100

ty 100
```

Input data = graph data * 10

Output:



Task 2: Scaling

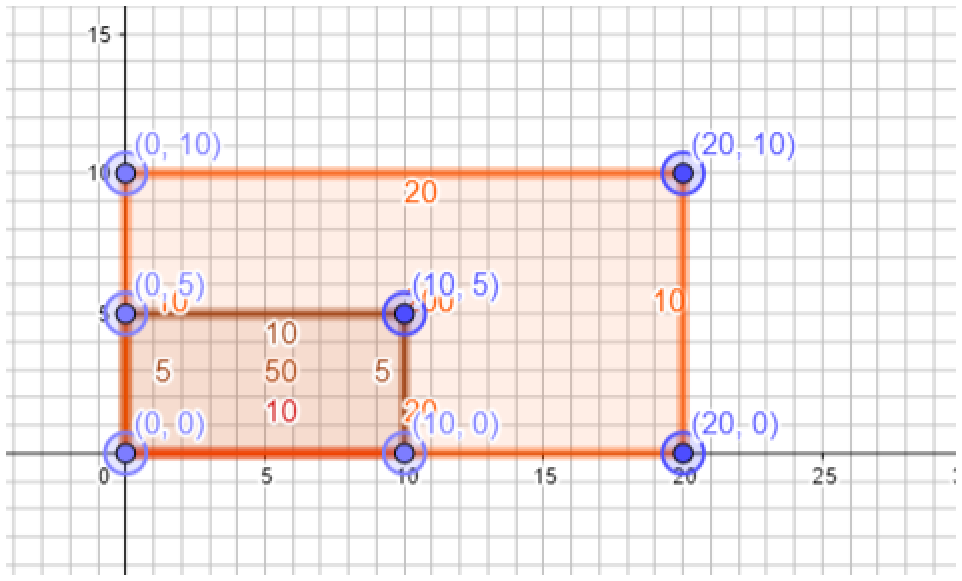
Code:

```
#include<windows.h>
#include <GL/glut.h>
#include <iostream>
using namespace std;

int ax, ay, bx, by, cx, cy, dx, dy, sx, sy;

void init(void)
{
      glClearColor(0.0, 0.0, 0.0, 0.0);

      glMatrixMode(GL_PROJECTION);

      gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
}

void drawShapes(void)
{
      glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1, 1, 0);
      glBegin(GL_QUADS);
         glVertex2i(ax*sx, ay*sy);
         glVertex2i(bx*sx, by*sy);
         glVertex2i(cx*sx, cy*sy);
         glVertex2i(dx*sx, dy*sy);
    glEnd();

    glColor3f(1, 0, 0);
      glBegin(GL_QUADS);
         glVertex2i(ax, ay);
         glVertex2i(bx, by);
         glVertex2i(cx, cy);
         glVertex2i(dx, dy);
    glEnd();
```

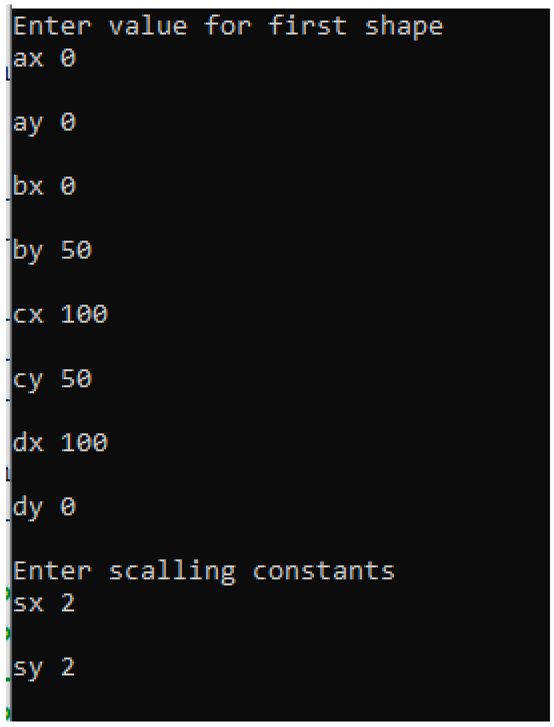
```
glFlush();

}
int main(int argc, char* argv[])
{

      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

      glutInitWindowPosition(1200, 100);
      glutInitWindowSize(700, 700);
      glutCreateWindow("Scalling");

      init();
      glutDisplayFunc(drawShapes);

      cout << "Enter value for first shape " << endl;
      cout << "ax ";
      cin >> ax;
      cout << endl;
      cout << "ay ";
      cin >> ay;
      cout << endl;

     cout << "bx ";
      cin >> bx;
      cout << endl;
      cout << "by ";
      cin >> by;
      cout << endl;

      cout << "cx ";
      cin >> cx;
      cout << endl;
      cout << "cy ";
      cin >> cy;
      cout << endl;
```

```
   cout << "dx ";
    cin >> dx;
    cout << endl;
    cout << "dy ";
    cin >> dy;
    cout << endl;


    cout << "Enter scalling constants " << endl;
    cout << "sx ";
    cin >> sx;
    cout << endl;

    cout << "sy ";
    cin >> sy;
    cout << endl;


    glutMainLoop();

    return 0;
}
```

Graph:



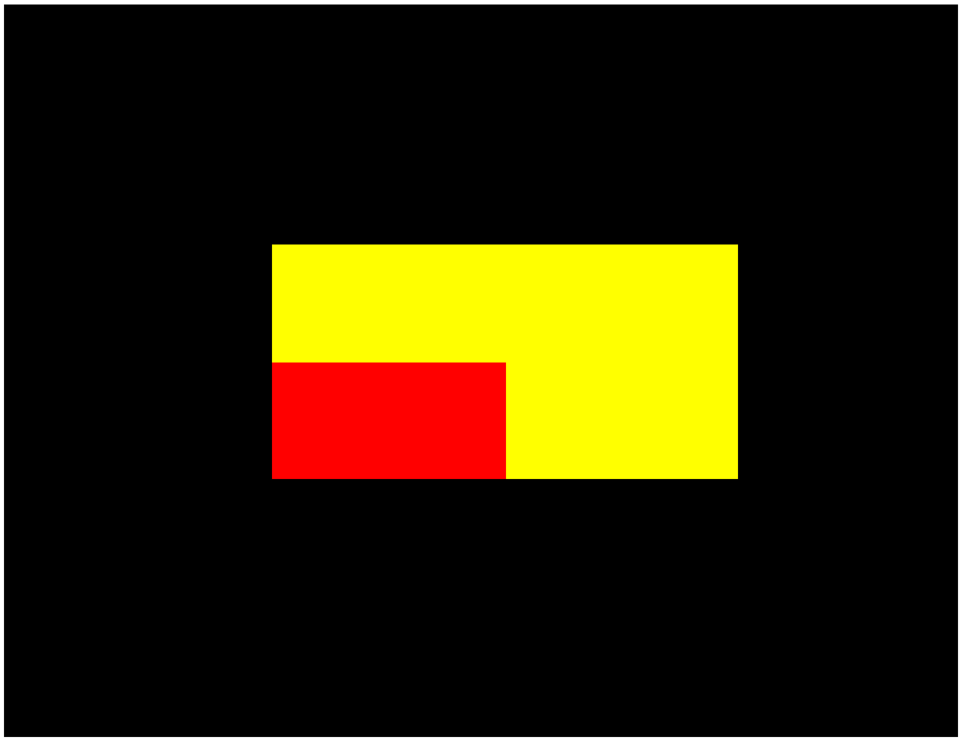
Input:



```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Enter scalling constants
sx 2

sy 2
```

Input data = graph data * 10

Output:

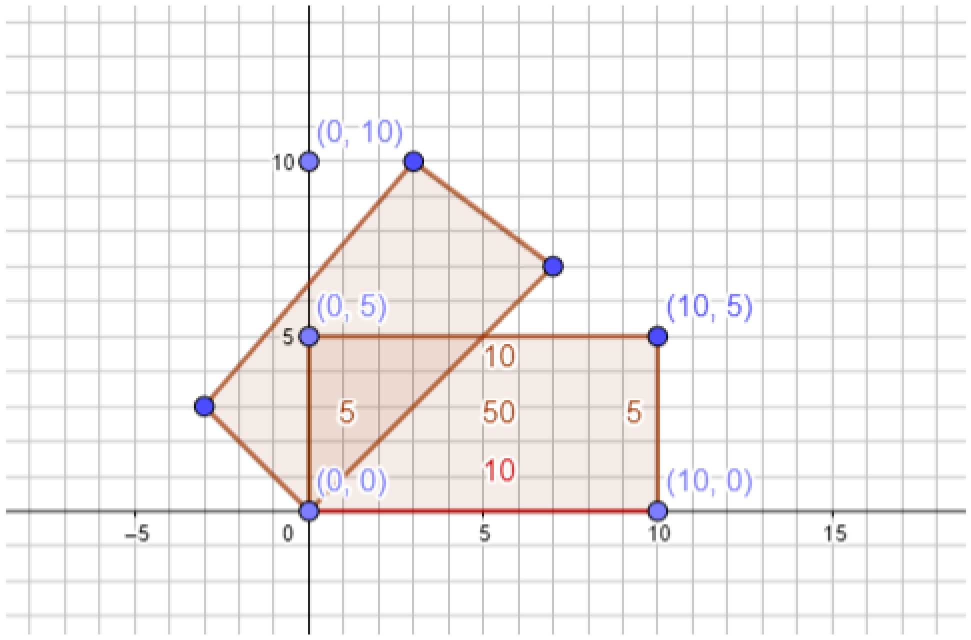


Task 3: Rotation

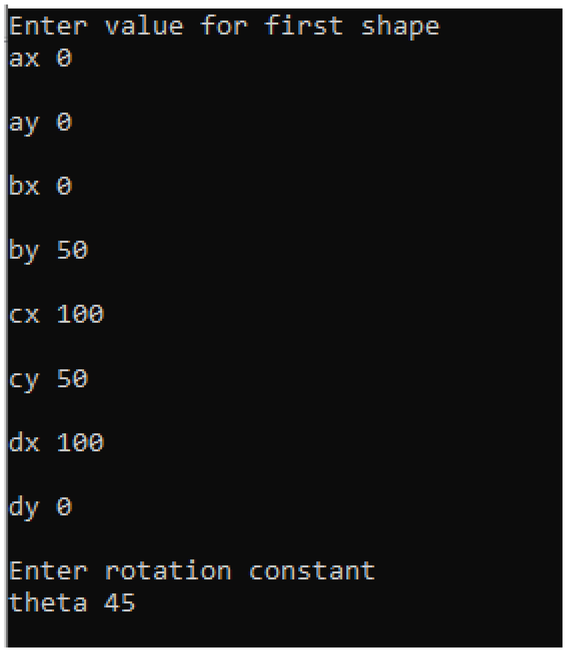
Code:

```
#include<windows.h>
#include <GL/glut.h>
#include <iostream>
#include <cmath>
using namespace std;

int ax, ay, bx, by, cx, cy, dx, dy, theta;
#define PI acos(-1.0)

void init(void)
{
      glClearColor(0.0, 0.0, 0.0, 0.0);

      glMatrixMode(GL_PROJECTION);

      gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
}

void drawShapes(void)
{
      glClear(GL_COLOR_BUFFER_BIT);


    glColor3f(1, 0, 0);
      glBegin(GL_QUADS);
        glVertex2i(ax, ay);
        glVertex2i(bx, by);
        glVertex2i(cx, cy);
        glVertex2i(dx, dy);
    glEnd();

    double r = PI*(theta)/180.0;
    double aX, aY, bX, bY, cX, cY, dX, dY;
    aX = ax*cos(r) - ay*sin(r);
    aY = ax*sin(r) + ay*cos(r);
    bX = bx*cos(r) - by*sin(r);
    bY = bx*sin(r) + by*cos(r);
    cX = cx*cos(r) - cy*sin(r);
    cY = cx*sin(r) + cy*cos(r);
    dX = dx*cos(r) - dy*sin(r);
    dY = dx*sin(r) + dy*cos(r);

    glColor3f(1, 1, 0);
     glBegin(GL_QUADS);
        glVertex2i(aX, aY);
        glVertex2i(bX, bY);
        glVertex2i(cX, cY);
        glVertex2i(dX, dY);
    glEnd();

glFlush();

}
int main(int argc, char* argv[])
{

     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

     glutInitWindowPosition(1200, 100);
     glutInitWindowSize(700, 700);
     glutCreateWindow("Rotation");

     init();
     glutDisplayFunc(drawShapes);

     cout << "Enter value for first shape " << endl;
     cout << "ax ";
     cin >> ax;
     cout << endl;
     cout << "ay ";
     cin >> ay;
     cout << endl;
    cout << "bx ";
     cin >> bx;
     cout << endl;
     cout << "by ";
     cin >> by;
     cout << endl;

     cout << "cx ";
     cin >> cx;
     cout << endl;
     cout << "cy ";
     cin >> cy;
     cout << endl;

    cout << "dx ";
     cin >> dx;
     cout << endl;
     cout << "dy ";
     cin >> dy;
     cout << endl;


     cout << "Enter rotation constant " << endl;
     cout << "theta ";
     cin >> theta;
     cout << endl;



     glutMainLoop();

     return 0;
}
```

Graph:



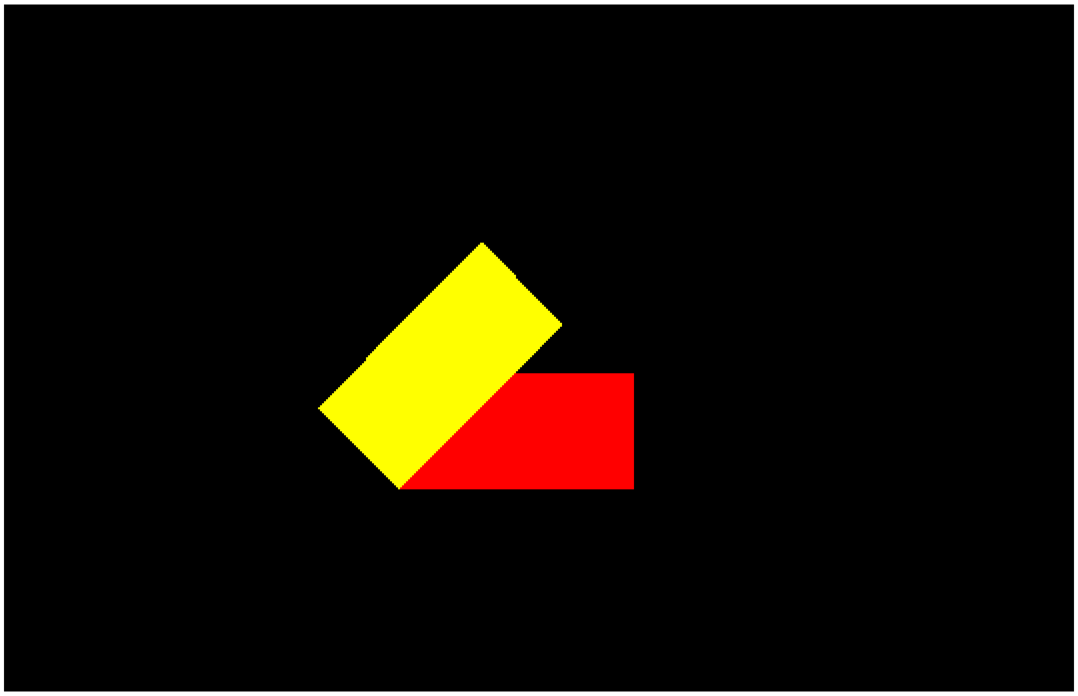
Input:



```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Enter rotation constant
theta 45
```

Input data = graph data * 10
Rotated data fits to nearest integer value in graph

Output:

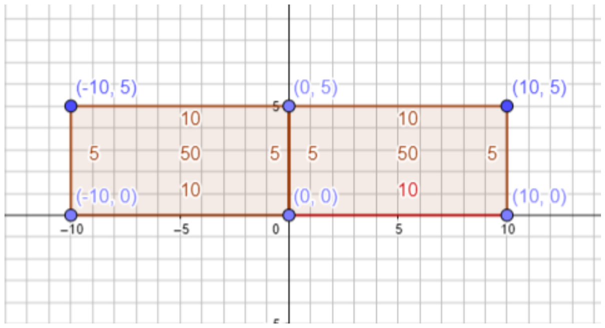
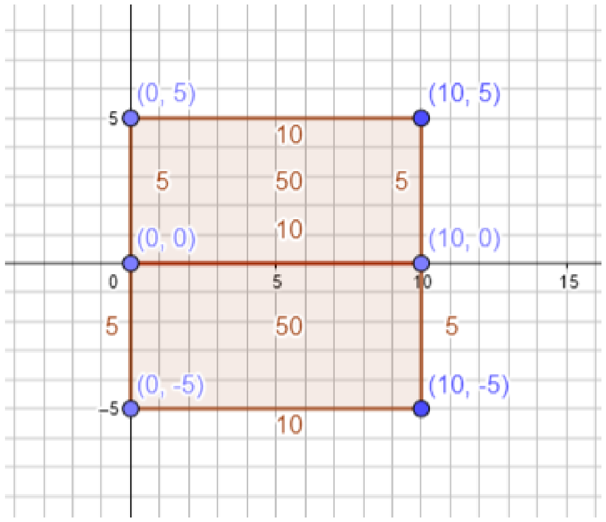


Task 1: Reflection

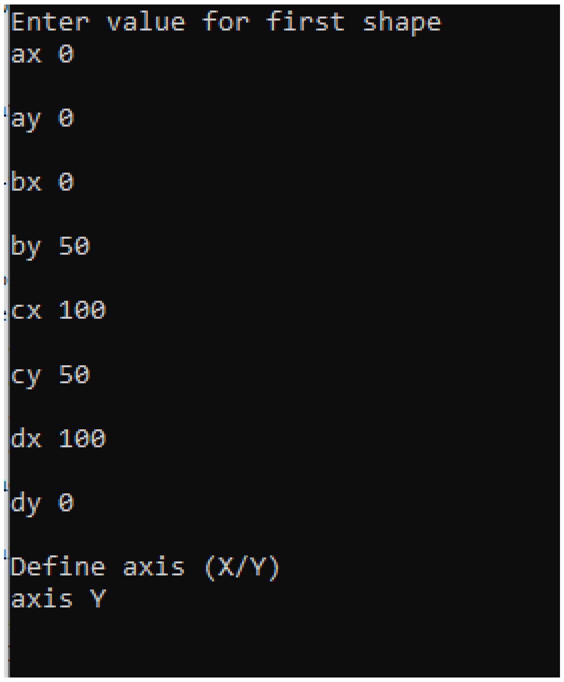
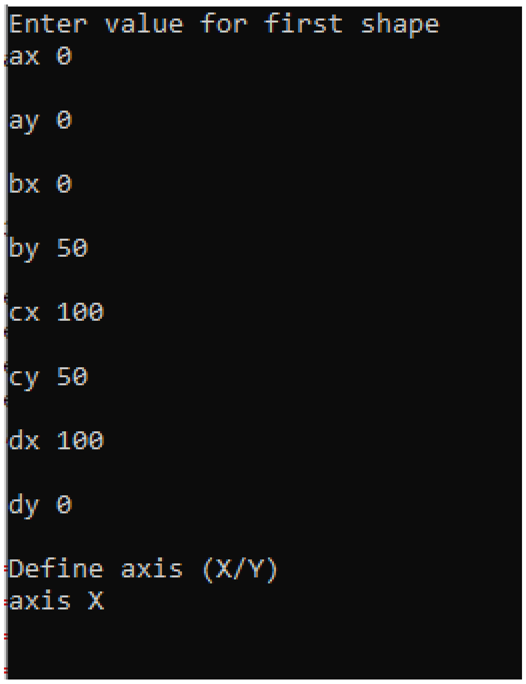
Code:

```
#include<windows.h>
#include <GL/glut.h>
#include <iostream>
#include <cmath>
using namespace std;

int ax, ay, bx, by, cx, cy, dx, dy;
char axis;

void init(void)
{
     glClearColor(0.0, 0.0, 0.0, 0.0);

     glMatrixMode(GL_PROJECTION);

     gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
}

void drawShapes(void)
{
     glClear(GL_COLOR_BUFFER_BIT);


    glColor3f(1, 0, 0);
     glBegin(GL_QUADS);
        glVertex2i(ax, ay);
        glVertex2i(bx, by);
        glVertex2i(cx, cy);
        glVertex2i(dx, dy);
    glEnd();

    if(axis == 'Y')
    {
        ax = ax*(-1);
        bx = bx*(-1);
        cx = cx*(-1);
        dx = dx*(-1);
    }
    else
    {
        ay = ay*(-1);
        by = by*(-1);
        cy = cy*(-1);
        dy = dy*(-1);
    }

    glColor3f(1, 1, 0);
      glBegin(GL_QUADS);
        glVertex2i(ax, ay);
        glVertex2i(bx, by);
        glVertex2i(cx, cy);
        glVertex2i(dx, dy);
    glEnd();

glFlush();

}
int main(int argc, char* argv[])
{

      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

      glutInitWindowPosition(1200, 100);
      glutInitWindowSize(700, 700);
      glutCreateWindow("Reflaction");

      init();
      glutDisplayFunc(drawShapes);

      cout << "Enter value for first shape " << endl;
      cout << "ax ";
      cin >> ax;
      cout << endl;
      cout << "ay ";
      cin >> ay;
      cout << endl;

    cout << "bx ";
      cin >> bx;
      cout << endl;
      cout << "by ";
      cin >> by;
      cout << endl;

      cout << "cx ";
      cin >> cx;
      cout << endl;
      cout << "cy ";
      cin >> cy;
      cout << endl;

    cout << "dx ";
      cin >> dx;
      cout << endl;
      cout << "dy ";
      cin >> dy;
      cout << endl;


      cout << "Define axis (X/Y) " << endl;
      cout << "axis ";
      cin >> axis;
      cout << endl;



      glutMainLoop();

      return 0;
}
```

Graph:

On Y-axis	On X-axis
	

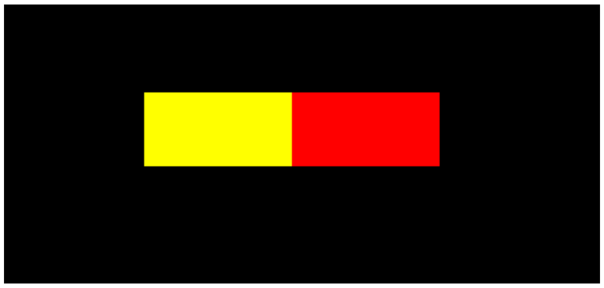
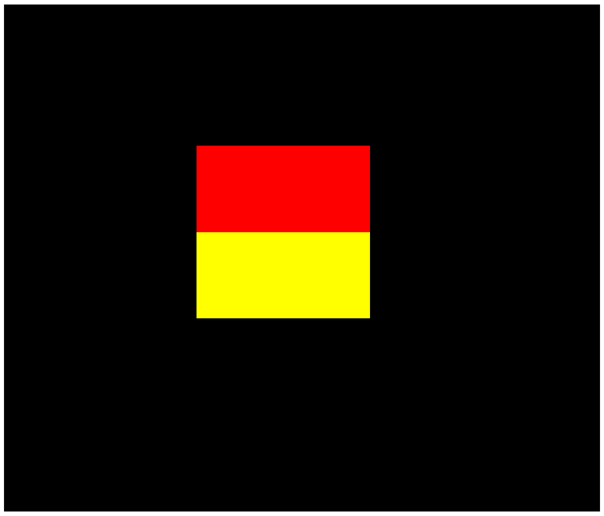
Input:

On Y-axis	On X-axis
	

```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Define axis (X/Y)
axis Y
```

```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Define axis (X/Y)
axis X
```

Output:

On Y-axis	On X-axis
	

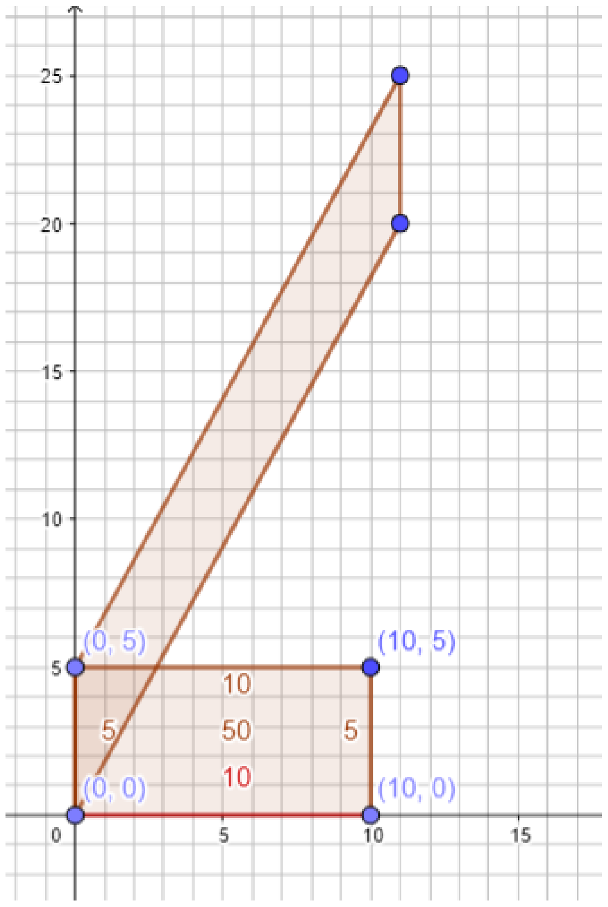
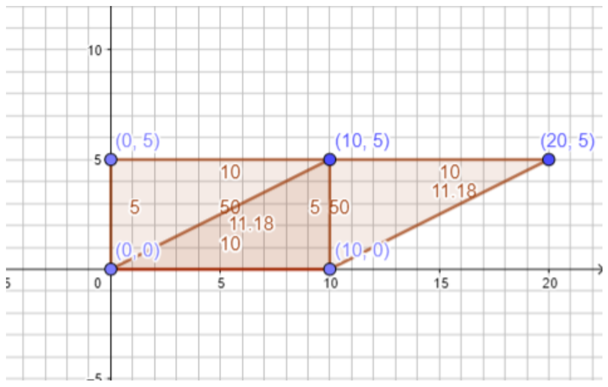
Task 1: Shearing

Code:

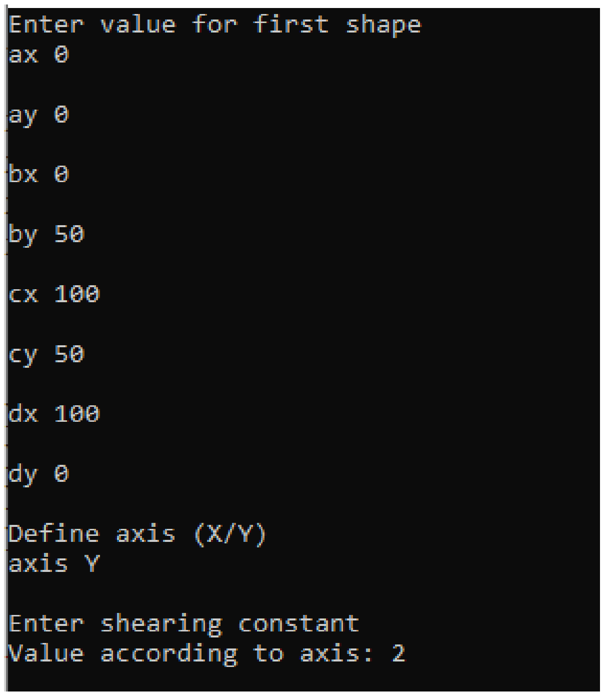
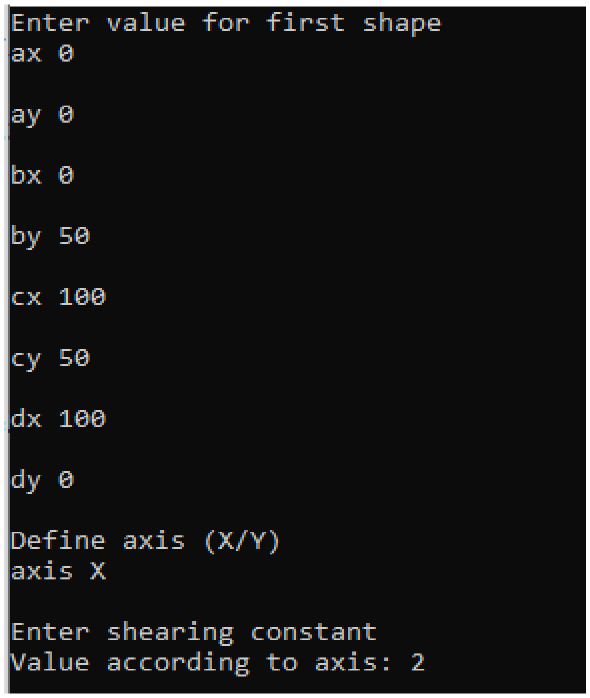
```
#include<windows.h>
#include <GL/glut.h>
#include <iostream>
using namespace std;

int ax, ay, bx, by, cx, cy, dx, dy, sh;
char axis;

void init(void)
{
      glClearColor(0.0, 0.0, 0.0, 0.0);

      glMatrixMode(GL_PROJECTION);

      gluOrtho2D(-300.0, 300.0, -300.0, 300.0);
}

void drawShapes(void)
{
      glClear(GL_COLOR_BUFFER_BIT);


    glColor3f(1, 0, 0);
      glBegin(GL_QUADS);
         glVertex2i(ax, ay);
         glVertex2i(bx, by);
         glVertex2i(cx, cy);
         glVertex2i(dx, dy);
    glEnd();

    if(axis == 'X')
    {
        ax = ax + ay*sh;
        bx = bx + by*sh;
        cx = cx + cy*sh;
        dx = dx + dy*sh;
    }
    else
    {
        ay = ay + ax*sh;
        by = by + bx*sh;
        cy = cy + cx*sh;
        dy = dy + dx*sh;
    }

    glColor3f(1, 1, 0);
     glBegin(GL_QUADS);
        glVertex2i(ax, ay);
        glVertex2i(bx, by);
        glVertex2i(cx, cy);
        glVertex2i(dx, dy);
    glEnd();

glFlush();

}
int main(int argc, char* argv[])
{

     glutInit(&argc, argv);
     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

     glutInitWindowPosition(1200, 100);
     glutInitWindowSize(700, 700);
     glutCreateWindow("Shearing");

     init();
     glutDisplayFunc(drawShapes);

     cout << "Enter value for first shape " << endl;
     cout << "ax ";
     cin >> ax;
     cout << endl;
     cout << "ay ";
```

```
      cin >> ay;
      cout << endl;

    cout << "bx ";
      cin >> bx;
      cout << endl;
      cout << "by ";
      cin >> by;
      cout << endl;

      cout << "cx ";
      cin >> cx;
      cout << endl;
      cout << "cy ";
      cin >> cy;
      cout << endl;

    cout << "dx ";
      cin >> dx;
      cout << endl;
      cout << "dy ";
      cin >> dy;
      cout << endl;


      cout << "Define axis (X/Y) " << endl;
      cout << "axis ";
      cin >> axis;
      cout << endl;

      cout << "Enter shearing constant " << endl;
      cout << "Value according to axis: ";
      cin >> sh;
      cout << endl;

      glutMainLoop();

      return 0;
}
```

Graph:

On Y-axis	On X-axis
	

Input:

On Y-axis	On X-axis
	

```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Define axis (X/Y)
axis Y

Enter shearing constant
Value according to axis: 2
```

```
Enter value for first shape
ax 0

ay 0

bx 0

by 50

cx 100

cy 50

dx 100

dy 0

Define axis (X/Y)
axis X

Enter shearing constant
Value according to axis: 2
```

Output:

On Y-axis	On X-axis